Leveraging Contextual Information from Function Call Chains to Improve Fault Localization

Árpád Beszédes[1], Ferenc Horváth[1], Massimiliano Di Penta[2], Tibor Gyimóthy[1]
[1] {beszedes,hferenc,gyimothy}@inf.u-szeged.hu, [2] dipenta@unisannio.it

Spectrum-Based Fault Localization (SBFL)

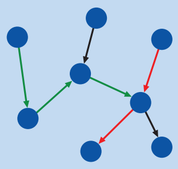
Code elements are more suspicious to contain a fault that are exercised by comparably more failing test cases than passing ones, while non-suspicious elements are traversed mostly by passing tests.

Motivation

- Debugging is a very cumbersome process.
- SBFL limits: only hit-based in most cases, which neglects any contextual information.
- Lot of research with various algorithms, but only marginal improvements.

Context

Chains can show, for instance, that a function may fail if called from one place and perform successfully when called from another.



Approach

- An SBFL algorithm computes ranking on all occurring call chains during execution.
- Ranked chains can be used at this point in debugging.
- Or, the most suspicious functions can be computed from the chain ranks (two strategies are available).

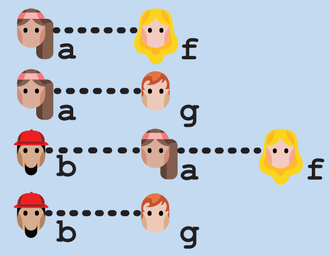
Call chains

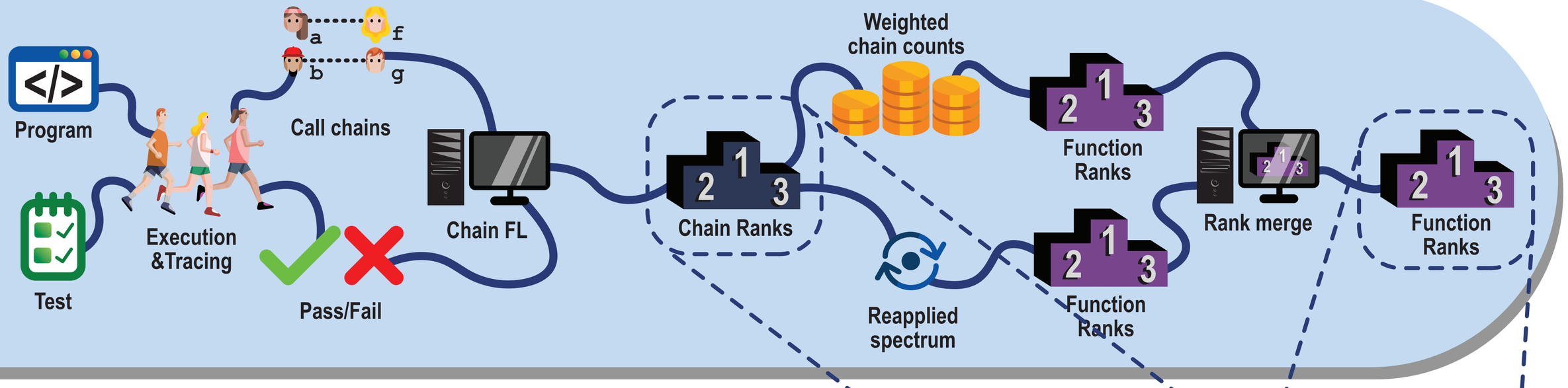
*Function call chains are snapshots of the call stack occurring during execution and as such can provide valuable **context** to the fault being traced. Call chains are artifacts which are well known to programmers who perform debugging.*

```
public class C {
  public void a(int i) {
    if (i < 0)
      f(i);
    else g(i);
  }
  public void b(int i) {
    if (i < 0)
      a(i);
    else g(i);
  }
  public void f(int i) {
... }
  public void g(int i) {
... }
}
```

```
thread "main" stack trace
at ex.C.g(C.java:20)
at ex.C.b(C.java:14)
at ex.M.main(M.java:7)

thread "main" stack trace
ex.C.f(C.java:17)
ex.C.a(C.java:6)
ex.C.b(C.java:12)
ex.M.main(M.java:8)
```





Chain	Rank
b→g	1
a→g	2
a→f	3
b→a→f	4
b	4

Function	Rank
g	1
a	2
b	3
f	4

Results

- Effectiveness improvement of Ochiai of 1 to 6 positions on average (relative improvement about **45%**).
- **76%** (33) of the defects with ranks >10 could be reduced to <10 (25) with average reduction of **84%**.

Program	Bugs	Ochiai rank	Comb. rank	Diff.	Rel. change	Ochiai rank > *10*	Reduced to < *10*	Rel. impr.
Commons Lang	46	4.7	3.9	-0.8	-17%	6	4 (9%)	-15.1 (-67%)
Commons Math	74	8.7	3.8	-4.9	-56%	18	15 (20%)	-24.3 (-87%)
JFreeChart	18	5.3	3.4	-1.9	-36%	2	2 (11%)	-19.0 (-76%)
Joda-Time	23	13.4	7.8	-5.6	-42%	7	4 (17%)	-43.1 (-88%)
Total / Average	**161**	**7.8**	**4.3**	**-3.5**	**-45%**	**33**	**25 (16%)**	**-25.4 (-84%)**